Planning & Scheduling Assignment 4 Report

Patrik Kupco

June 9, 2025

Name: Patrik Kupco

Registration Number: ies25210

Email: kupco.patrik.16@gmail.com

1 Exercise 1: Truck Deliveries Problem

1.1 Problem Description

The truck deliveries problem involves driving boxes via various trucks capable of specific speeds, and weight capacities to their destinations, while respecting the weights of the boxes themselves and the limit of drivers in the company (2). The result is the minimum time required to deliver all of the boxes to their destination.

Key constraints:

- Trucks have a maximum weight capacity given by `max_truck_weight`, which has to be below the weight of a specific box.
- Trucks also have specific speeds given by `speed`, which is used to calculate the time required to deliver a box to its destination, which distance is given by `distance`
- The company only has 2 drivers, so only 2 trucks can be on delivery at any time.
- Each box becomes available to deliver after a time given by `ready_after`.

1.2 Solution Code

```
array[DELIVERIES] of var TRUCKS: onTruck;
array[DELIVERIES] of var TIME: depart;
array[DELIVERIES] of var int: delivery_time;
array[DELIVERIES] of var TIME: finish_time;
```

At the beginning of the code there are three decision variables to solve this problem. First one is the `onTruck`, which holds which truck will be used to deliver which box, `depart`, which holds the time at which the truck departs with a box for delivery and `delivery_time`, which holds the time it will take for the specific truck to get to its destination.

There is also the `finish_time` variable, which is not required, but it is used to hold the time at which the delivery finishes, so that the model keeps track of when a truck is available for another delivery.

1. Depart After Box Is Ready Constraint

```
constraint forall(d in DELIVERIES)(
  depart[d] >= ready_after[d]
);
```

This constraint defines that each truck can only depart with a box after it becomes ready to deliver.

2. Delivery and Finish Time Constraints

```
constraint forall(d in DELIVERIES)(
  delivery_time[d] = distance[d] div speed[onTruck[d]]
);

constraint forall(d in DELIVERIES)(
  finish_time[d] = depart[d] + delivery_time[d]
);
```

These two constraints define the time which it will take for the box to be delivered by truck to its destination, and the exact time, at which the delivery will be delivered.

3. Max Weight Constraint

```
constraint forall(d in DELIVERIES)(
  weight[d] <= max_truck_weight[onTruck[d]]
);
```

This constraints defines the boxes each truck is capable of delivering with respect to its max weight limit.

4. Avoid Overlapping Deliveries on the Same Truck Constraint

```
constraint forall(d1, d2 in DELIVERIES where d1 < d2)(
  onTruck[d1] = onTruck[d2] ->
  (finish_time[d1] <= depart[d2] \/ finish_time[d2] <= depart[d1])
);
```

This constraint ensures that two deliveries assigned to the same truck do not overlap in time. If two deliveries are on the same truck, one must finish before the other starts. This avoids scheduling conflicts and makes sure that a truck can only handle one delivery at a time.

5. Maximum Available Drivers Constraint

```
constraint cumulative(depart, delivery_time, [1 | d in DELIVERIES], 2);
```

This constraint ensures that there are only 2 trucks on a delivery at the same time, since only 2 drivers work for the given company.

Objective Definition

```
var TIME: makespan;
constraint makespan = max(finish_time);
solve minimize makespan;
```

This constraint is responsible for choosing the maximum finish time of a delivery, and defines that the model should optimize the solution to find a minimum time for this problem.

1.3 Execution Examples

1.3.1 Example 1

Input Data:

```
ntrucks = 3;
max_truck_weight = [15,20,30];
speed = [10,20,5] ;

% Number of deliveries and the associated set
ndeliveries = 5;
weight = [5,10,18,22,10];
ready_after = [0,0,3,2,1];
distance = [40,100,60,80,120];
```

Output:

```
depart = [23, 13, 18, 2, 1];
 delivery_time = [4, 10, 3, 16, 12];
 onTruck = [1, 1, 2, 3, 1];
 makespan = 27
----------
 depart = [22, 0, 18, 2, 10];
 delivery_time = [4, 10, 3, 16, 12];
```

```
 onTruck = [1, 1, 2, 3, 1];
 makespan = 26
----------
 depart = [0, 0, 3, 6, 10];
 delivery_time = [2, 10, 3, 16, 12];
 onTruck = [2, 1, 2, 3, 1];
 makespan = 22
----------
 depart = [0, 0, 17, 4, 5];
 delivery_time = [4, 5, 3, 16, 12];
 onTruck = [1, 2, 2, 3, 1];
 makespan = 20
----------
 depart = [0, 0, 10, 2, 13];
 delivery_time = [2, 10, 3, 16, 6];
 onTruck = [2, 1, 2, 3, 2];
 makespan = 19
----------
 depart = [5, 0, 13, 2, 7];
 delivery_time = [2, 5, 3, 16, 6];
 onTruck = [2, 2, 2, 3, 2];
 makespan = 18
----------
==========
```

1.3.2 Example 2

Input Data:

```
ntrucks = 4;
max_truck_weight = [15,20,40,25];
speed = [20,5,10,10] ;

% Number of deliveries and the associated set
ndeliveries = 9;
weight = [30,10,18,22,10,15,35,10,15];
ready_after = [0,0,3,2,1,2,3,1,1];
distance = [40,100,60,80,120,60,140,20,140];
```

Output:

```
 depart = [25, 28, 3, 17, 22, 33, 3, 1, 15];
 delivery_time = [4, 5, 12, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 2, 3, 1, 1, 3, 1, 1];
 makespan = 36
----------
 depart = [27, 7, 15, 17, 1, 12, 3, 32, 25];
 delivery_time = [4, 5, 12, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 2, 3, 1, 1, 3, 1, 1];
 makespan = 33
----------
 depart = [28, 7, 16, 17, 1, 12, 3, 15, 25];
 delivery_time = [4, 5, 12, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 2, 3, 1, 1, 3, 1, 1];
 makespan = 32
----------
 depart = [0, 26, 18, 18, 8, 14, 4, 17, 1];
 delivery_time = [4, 5, 12, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 2, 3, 1, 1, 3, 1, 1];
 makespan = 31
----------
 depart = [0, 0, 18, 18, 12, 26, 4, 29, 5];
 delivery_time = [4, 5, 12, 8, 6, 3, 14, 1, 7];
```

```
 onTruck = [3, 1, 2, 3, 1, 1, 3, 1, 1];
 makespan = 30
----------
 depart = [25, 0, 5, 17, 18, 24, 3, 27, 11];
 delivery_time = [4, 5, 6, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 4, 3, 1, 1, 3, 1, 1];
 makespan = 29
----------
 depart = [0, 0, 5, 18, 21, 18, 4, 27, 11];
 delivery_time = [4, 5, 6, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 4, 3, 1, 1, 3, 1, 1];
 makespan = 28
----------
 depart = [0, 0, 21, 18, 15, 12, 4, 26, 5];
 delivery_time = [4, 5, 6, 8, 6, 3, 14, 1, 7];
 onTruck = [3, 1, 4, 3, 1, 1, 3, 1, 1];
 makespan = 27
----------
==========
```

2 Exercise 2: Flight Control Center Problem

2.1 Problem Description

The flight control center problem involves scheduling daily shifts (morning, afternoon, night, off) for a number of air traffic controllers over a 7-day period.

Each shift has an associated workload in units: 1 for morning, 2 for afternoon, and 4 for night. Controllers must rest (take a day off) after reaching 4 workload units. Each controller is also required to work at least one of each shift type (morning, afternoon, night) during the week.

Additionally, each day must be staffed with at least 3 morning, 2 afternoon, and 1 night shift controllers. The goal is to generate a valid shift schedule that minimizes the difference (gap) between the maximum and minimum total workload among the controllers.

Key constraints:

- Each controller must rest (OFF) after accumulating 4 workload units, regardless of the combination of shifts.
- The weekly schedule spans 7 days, with exactly one shift (or OFF) assigned to each controller per day.
- Shift unit values are: MORNING = 1 unit, AFTERNOON = 2 units, NIGHT = 4 units, OFF = 0 units.
- Minimum daily staffing requirements:
  - At least 3 controllers on the MORNING shift
  - At least 2 controllers on the AFTERNOON shift
  - At least 1 controller on the NIGHT shift
- Each controller must work at least one MORNING, one AFTERNOON, and one NIGHT shift during the week.
- The objective is to minimize the `gap` — the difference between the maximum and minimum total workload units among all controllers.

2.2 Solution Code

1. Finite State Machine Constraint

```
int: ns = 5;
set of int : STATE = 1..ns;
array[STATE,CONTROL_SHIFT] of opt STATE : allowed =
 % MRN AFT NGT OFF
 [| 2,  3,  5, 1
  | 3,  4, <>, 1
  | 4,  5, <>, 1
  | 5, <>, <>, 1
  |<>, <>, <>, 1|];

constraint forall(c in CONTROLLERS)(
  regular(row(roster,c), allowed, 1, STATE)
);
```

This part of the code defines the Finite State Machine, that holds all possible states a controller can be in (states representing how many workload units they currently have). A controller gains 1 point for each morning shift, 2 points for each afternoon shift and 4 points for

each night shift. After reaching 4 points, the controller is required to take a break, and also it shouldn't be possible for the controller to have more than 4 points accumulated.

2. Daily Staffing Requirements Constraint

```
constraint forall(d in DAYS) (
  sum(c in CONTROLLERS)(roster[c,d] = MORNING) >= 3
  /\
  sum(c in CONTROLLERS)(roster[c,d] = AFTERNOON) >= 2
  /\
  sum(c in CONTROLLERS)(roster[c,d] = NIGHT) >= 1
);
```

This constraint describes the minimum amount of controllers on each shift on the given day (3 for the MORNING, 2 for AFTERNOON and 1 for NIGHT)

3. Shift Coverage Constraint

```
constraint forall(c in CONTROLLERS) (
  sum(d in DAYS)(roster[c,d] = MORNING) >= 1
  /\
  sum(d in DAYS)(roster[c,d] = AFTERNOON) >= 1
  /\
  sum(d in DAYS)(roster[c,d] = NIGHT) >= 1
);
```

This constraint ensures that each air traffic controller does their mandatory shift count for each type of shift, that being once for each (morning, afternoon, night).

4. Workload Units Sum Constraint

```
array[CONTROL_SHIFT] of int: shift_work = [MORNING: 1, AFTERNOON: 2, NIGHT: 4,
    OFF: 0];

constraint forall(c in CONTROLLERS) (
work[c] = sum(d in DAYS)(shift_work[roster[c,d]])
);
```

This constraint calculates the total workload units for each air traffic controller based on their assigned shifts throughout the week. The `shift_work` array is used to associate each shift type with its corresponding workload value: 1 unit for MORNING, 2 units for AFTERNOON, 4 units for NIGHT, and 0 units for OFF. The resulting workload is stored in the `work` array, which is later used to calculate the `gap` between controllers.

5. Maximum Counts of Shifts Constraint

```
constraint forall(c in CONTROLLERS) (
  sum(d in DAYS)(roster[c,d] = NIGHT) <= 2
  /\
  sum(d in DAYS)(roster[c,d] = AFTERNOON) <= 3
  /\
  sum(d in DAYS)(roster[c,d] = MORNING) <= 3
);
```

In this constraint there are limits on how many shifts are possible within the given constraints of the exercise. This constraint is here for optimization purposes.

To achieve maximum amount of nights, a controller could work the following shift:

```
NIGHT OFF NIGHT OFF AFTERNOON MORNING MORNING
```

To achieve maximum amount of afternoons, a controller could work the following shift:

```
AFTERNOON AFTERNOON OFF AFTERNOON MORNING OFF NIGHT
```

And to achieve maximum amount of mornings, a controller could work the following shift:

```
MORNING MORNING MORNING OFF NIGHT OFF AFTERNOON
```

6. Max Workload Units Constraint

```
constraint forall(c in CONTROLLERS) (
  work[c] >= 7 /\ work[c] <= 12
);
```

This constraint is also here for optimization reasons, and gives out the range of workload units given the constraints in the Exercise. The minimum is for working one of each shifts only, to reach the maximum the controller could work the following shift

```
NIGHT OFF NIGHT OFF AFTERNOON MORNING MORNING
```

7. Max Workload Units Constraint

```
constraint forall(c in CONTROLLERS) (
  work[c] >= 7 /\ work[c] <= 12
);
```

This constraint is also here for optimization reasons, and gives out the range of workload units given the constraints in the Exercise. The minimum is for working one of each shifts only, to reach the maximum the controller could work the following shift

```
NIGHT OFF NIGHT OFF AFTERNOON MORNING MORNING
```

Objective Definition

```
constraint gap = max(work) - min(work);
constraint gap = 1;
solve
  :: int_search(
       [roster[c,d] | c in CONTROLLERS, d in DAYS],
       dom_w_deg,
       indomain_random,
       complete
     )
  :: restart_luby(1000)
  minimize gap;
```

At the end of the code is the constraint that is used to calculate the outcome of the problem (the plan). The model is supposed to hit a gap of 1, meaning that the difference between most overworked and most underworked controller will be only 1 workload unit.

The model uses certain heuristics to achieve this, specifically `dom_w_deg` for variable selection and `indomain_random` for value selection.

Most importantly, the model uses `restart_luby`, which is a strategy that restarts the search regularly using the Luby sequence. This helps the solver avoid getting stuck by trying different parts of the search space. The number 1000 means the solver will restart after a certain number of steps based on the Luby pattern. This increases the chances of finding a good solution.

From the given datafiles, this restart strategy doesnt have much effect on `air_traffic1.dzn`, however the solution time for `air_traffic0.dzn` goes from far too long to be relevant, to a matter of milliseconds

2.3 Execution Examples

2.3.1 Example 1

Input Data:

```
air_t_controllers = 10;
```

Output:

```
% Just for debugging. The table of shifts.
%         1         2         3         4         5         6         7
%    MORNING       OFF     NIGHT       OFF AFTERNOON   MORNING   MORNING
%    MORNING   MORNING AFTERNOON       OFF   MORNING       OFF     NIGHT
%  AFTERNOON       OFF   MORNING AFTERNOON       OFF     NIGHT       OFF
%      NIGHT       OFF   MORNING   MORNING       OFF   MORNING AFTERNOON
%  AFTERNOON   MORNING       OFF AFTERNOON       OFF     NIGHT       OFF
%    MORNING   MORNING       OFF   MORNING AFTERNOON       OFF     NIGHT
%        OFF     NIGHT       OFF   MORNING   MORNING AFTERNOON       OFF
%      NIGHT       OFF AFTERNOON       OFF   MORNING   MORNING   MORNING
%        OFF AFTERNOON   MORNING       OFF     NIGHT       OFF AFTERNOON
%        OFF AFTERNOON       OFF     NIGHT       OFF AFTERNOON   MORNING
% % Raw Data in Tables ------------
roster = [MORNING, OFF, NIGHT, OFF, AFTERNOON, MORNING, MORNING, MORNING,
    MORNING, AFTERNOON, OFF, MORNING, OFF, NIGHT, AFTERNOON, OFF, MORNING,
    AFTERNOON, OFF, NIGHT, OFF, NIGHT, OFF, MORNING, MORNING, OFF, MORNING,
    AFTERNOON, AFTERNOON, MORNING, OFF, AFTERNOON, OFF, NIGHT, OFF, MORNING,
    MORNING, OFF, MORNING, AFTERNOON, OFF, NIGHT, OFF, NIGHT, OFF, MORNING,
    MORNING, AFTERNOON, OFF, NIGHT, OFF, AFTERNOON, OFF, MORNING, MORNING,
    MORNING, OFF, AFTERNOON, MORNING, OFF, NIGHT, OFF, AFTERNOON, OFF, AFTERNOON
    , OFF, NIGHT, OFF, AFTERNOON, MORNING];
work = [9, 9, 9, 9, 9, 9, 8, 9, 9, 9];
gap=1;
----------
==========
```

2.3.2 Example 2

Input Data:

```
air_t_controllers = 12;
```

Output:

```
% Just for debugging. The table of shifts.
%         1         2         3         4         5         6         7
%  AFTERNOON   MORNING       OFF       OFF   MORNING       OFF     NIGHT
%      NIGHT       OFF       OFF       OFF   MORNING   MORNING AFTERNOON
%      NIGHT       OFF   MORNING   MORNING   MORNING       OFF AFTERNOON
%    MORNING AFTERNOON   MORNING       OFF     NIGHT       OFF   MORNING
%    MORNING   MORNING       OFF AFTERNOON       OFF     NIGHT       OFF
%  AFTERNOON AFTERNOON       OFF       OFF   MORNING       OFF     NIGHT
%        OFF     NIGHT       OFF       OFF   MORNING AFTERNOON   MORNING
%    MORNING   MORNING       OFF     NIGHT       OFF       OFF AFTERNOON
%    MORNING       OFF   MORNING   MORNING AFTERNOON       OFF     NIGHT
%      NIGHT       OFF AFTERNOON   MORNING       OFF AFTERNOON       OFF
%        OFF       OFF     NIGHT       OFF AFTERNOON   MORNING   MORNING
%      NIGHT       OFF AFTERNOON AFTERNOON       OFF   MORNING       OFF
% % Raw Data in Tables ------------
roster = [AFTERNOON, MORNING, OFF, OFF, MORNING, OFF, NIGHT, NIGHT, OFF, OFF,
    OFF, MORNING, MORNING, AFTERNOON, NIGHT, OFF, MORNING, MORNING, MORNING, OFF
    , AFTERNOON, MORNING, AFTERNOON, MORNING, OFF, NIGHT, OFF, MORNING, MORNING,
     MORNING, OFF, AFTERNOON, OFF, NIGHT, OFF, AFTERNOON, AFTERNOON, OFF, OFF,
    MORNING, OFF, NIGHT, OFF, NIGHT, OFF, OFF, MORNING, AFTERNOON, MORNING,
```

```
    MORNING, MORNING, OFF, NIGHT, OFF, OFF, AFTERNOON, MORNING, OFF, MORNING,
    MORNING, AFTERNOON, OFF, NIGHT, NIGHT, OFF, AFTERNOON, MORNING, OFF,
    AFTERNOON, OFF, OFF, OFF, NIGHT, OFF, AFTERNOON, MORNING, MORNING, NIGHT,
    OFF, AFTERNOON, AFTERNOON, OFF, MORNING, OFF];
work = [8, 8, 9, 9, 8, 9, 8, 8, 9, 9, 8, 9];
gap=1;
----------
==========
```